THE SEARCH FOR STATISTICAL INDISTINGUISHABILITY BETWEEN REAL AND FAKE IMAGES

Anonymous authors
Paper under double-blind review

ABSTRACT

Images produced by generative models appear strikingly similar to real ones. In this work, we study whether perfection has been reached by generated content in terms of matching the properties of real data. We train deep neural networks to separate real data from fake and observe their test accuracies. Only a chance performance has any scope of proving statistical indistinguishability. However, since generative models are known to sometimes introduce small errors, e.g., low level checkerboard artifacts, we create progressive abstractions of the original real/fake images to potentially bring them to a space where those artifacts get wiped away. These include operations such as blurring, downsizing, cropping, converting RGB images into edge/semantic maps, etc. To our surprise, almost none of the abstractions are able to make fake data indistinguishable from real. The only cases when neural network based classifiers fail to separate the two categories is when the data is abstracted to such an extent that the original content seems mostly lost to the human eye, e.g., downsizing a 256×256 RGB image of human face to 4×4 resolution. We observe this trend across four different datasets of varying complexity – MNIST, CelebA, FFHQ and GenImage – and across various types of generative models – GANs, VAEs, diffusion models. This shows that the nature of fake artifacts extends all the way from low level to very high level in images. The simple implication is that fake images are much more different from real than we are used to think.

1 INTRODUCTION

The goal of an image generative model is to produce realistic images from a target distribution. Perfection at this task would mean that the generated images are *indistinguishable* from real ones. Recent generative models have seemingly made a lot of progress towards this goal (Midjourney, 2023; Ramesh et al., 2022). Typically, this idea of *indistinguishability* is limited to human judgment, where a human person should not be able to identify which image is real and which one is fake. However, there is an even more challenging test available for these models to pass successfully. The generated images and real images need to be indistinguishable not just to humans, but also to a neural network based classifier trained to distinguish them, i.e., its performance on the binary real-vs-fake classification task should stay at chance-level ($\sim 50\%$). This is known as the two-sample test (Lopez-Paz & Oquab, 2017). In this work, we study the extent to which generative models have come close to passing this ultimate test.

If we consider the real and fake image data in its original RGB space, a classifier can separate them easily. For example, in Fig. 1 column 1, a classifier achieves close to 100% test accuracy in distinguishing real human faces (Karras et al., 2019) from those generated by StyleGAN (Karras et al., 2019). This ease of separation exists not just for GANs but also for diffusion models (Cozzolino et al., 2024). In short, no generative model can pass this test in the RGB space (yet). The common reason for this high accuracy is that fake images have certain isolated properties, e.g., presence of low-level checkerboard artifacts (Zhang et al., 2019; Wang et al., 2020) which the classifier utilizes to make its determination. However, these artifacts often remain imperceptible to the human eye. In fact, the overall quality of generated images has become visually so convincing that humans are struggling to identify what is real and what is fake (Lu et al., 2023; Diel et al., 2024). Could it be that these models are failing the test only because of low level artifacts, but for higher level properties – e.g., outline of a face, positioning of facial keypoints – these models are actually close to passing the two

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

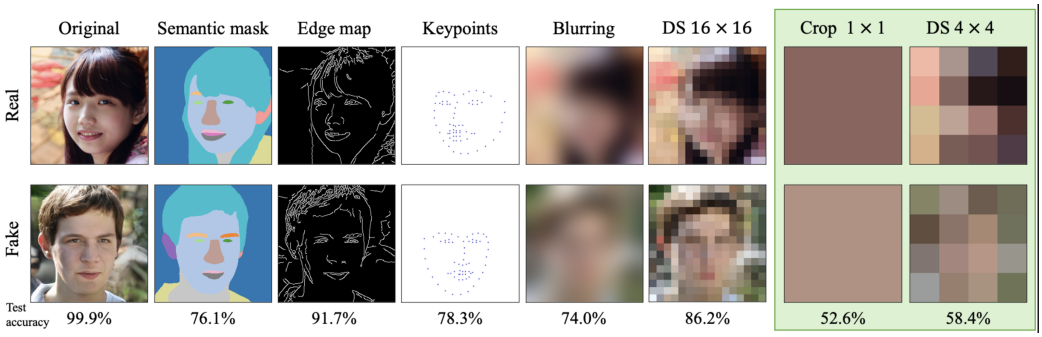


Figure 1: The first column corresponds to a classifier being trained to distinguish clean real (top) and fake images (bottom), achieving a high test accuracy because of the presence of artifacts. In each subsequent column, we transform the original images with the hope of masking away the artifact signal; e.g., downsizing (DS) to 16×16 (6th column). When new classifiers are trained for each column, the accuracy still remains high. For true *statistical indistinguishability* ($\sim 50\%$; last two columns), we need to abstract away most of the image content by, e.g., downsizing to 4×4 resolution.

sample test? The question that we study, then, is how much we need to *ignore* to reach that ultimate goal where even a neural network based classifier cannot distinguish real from fake. To study this, we create various levels of data abstractions where the original raw pixels of real and fake images are transformed into a representation space where some information gets lost. These operations include downsampling, cropping, blurring, converting into edge/semantic maps etc. For each of these abstractions, we train new classifiers to see if the transformed real and fake data are now separable. In doing so, we observe an unexpected phenomenon. In almost all cases, the classifier is able to obtain a high test accuracy (significantly above 50%). Fig. 1 shows some examples of these data abstractions and corresponding accuracies. It turns out that the only means by which the classifier attains close to 50% test accuracy is if we abstract away *almost the entire information* localized in the original real and fake images; e.g., downsizing 256×256 resolution images to 4×4 (last column).

We conduct this study on a variety of real image datasets including MNIST (LeCun, 1998), CelebA (Liu et al., 2015), FFHQ (Karras et al., 2019) and GenImage (Zhu et al., 2023), and fake images taken from various types of generative models like ACGAN (Odena et al., 2017), DCGAN (Radford et al., 2016), StyleGAN2 (Karras et al., 2020), VD VAE (Child, 2021) and text-to-image diffusion models like Midjourney, Stable Diffusion (wuk, 2022; Rombach et al., 2022b; Midjourney, 2023). Across all these settings, the general trend holds in which the statistical indistinguishability, i.e., $\sim 50\%$ test accuracy cannot be achieved until the representation level has abstracted away most of the information available in the input. We believe that our results should be surprising as they point out that the divide between real images and those produced by generative models is bigger than we typically imagine.

2 Related Work

Generative model evaluation. The field of generative models has been transformed in the last decade. From the original models which could barely generate handwritten digits (Goodfellow et al., 2014; Kingma & Welling, 2013), we have come a long way where models can now generate almost anything, be it art, realistic renderings of natural scenes (Rombach et al., 2022b; Ramesh et al., 2022; 2021; Rombach et al., 2022a; Midjourney, 2023). Many metrics have been proposed which try to capture properties like image quality and diversity of the generated images. Inception score Salimans et al. (2016) and FID Heusel et al. (2017) were one of the first popular metrics which encapsulate both of these aspects into one score. Other metrics were proposed which had the objective of disentangling these two aspects, to know how exactly a particular model is failing, or to have a better feature space than FID (Kynkäänniemi et al., 2019; Sajjadi et al., 2018; Alaa et al., 2022; Stein et al., 2023). Another way to evaluate how close generated images are to real ones is through two sample test (Lopez-Paz & Oquab, 2017), where a classifier is trained to distinguish the two classes; if it cannot succeed at this task (accuracy remaining $\sim 50\%$), then it can be inferred that the fake and real images come from exactly the same underlying distribution. For most generative models,

this test in its original form always gives a very high accuracy (e.g., Fig. 1 column 1). However, just because a generative model did not approximate the entirety of real image distribution perfectly does not necessarily imply that it cannot perfectly model *some* aspect of that distribution. E.g., maybe the real and fake images are completely indistinguishable when considering only the edge details of the faces, or locations of important keypoints. In this work, we employ the classifier test to study if perfect statistical indistinguishability been achieved on some meaningful representations of images.

Properties of fake images and detection. The image forensics community has tried to study if there are properties of fake images that can be used to distinguish them from real. Some have shown that the commonly used upsampling operations in a generative model often leave a high frequency trace in the generated image which can sometimes be seen with naked eyes as well (Zhang et al., 2019; Odena et al., 2016; Ma et al., 2025). Some works have shown where, spatially speaking, these artifacts reside most, e.g., in a person's hair (Chai et al., 2020). Some works have used other higher level features like turning off high frequency component, representations from pretrained encoder, geometric properties of the scene (Sarkar et al., 2024; Mandelli et al., 2026; Ojha et al., 2024). All of these different observations have then been used to deploy fake image detectors which can detect images not just from the model used during training, but also from unseen generative models (Wang et al., 2020; Cozzolino et al., 2019; Ojha et al., 2024; Cozzolino et al., 2024; Rajan et al., 2025; Wang et al., 2023). Our work is also concerned with studying the properties of fake images. However, the principal way in which our goal differs is that we wish to test if the errors in generative models are only due to certain isolated sources (e.g., checkerboard artifacts), or are they spread throughout in a much more holistic way. Furthermore, to make these detectors work in real world settings, past work has studied the sensitivity of trained detectors to different image manipulations and degradation operations like blurring, cropping, JPEG compression (Wang et al., 2020; Rajan et al., 2025; Yermakov et al., 2026). This has been aided in part by standardized datasets that test a detector's robustness to various post processing operations in a systematic way (Yan et al., 2023; Li et al., 2025). The general conclusion from these works is that increasing the strength of image manipulation (e.g., blurring) hurts the detector's performance. While our work also deals with similar image manipulations, it serves a very different purpose. Our goal is not to build a single all-purpose classifier that can detect real/fake images from all representation level. Rather, we train *new* classifiers from scratch for each representation level (different columns of Fig. 1) to see if the real and fake images becomes statistically indistinguishable at any such level.

3 Approach

We will first briefly formalize what data is, using the example of MNIST. Let $\mathcal{R}$ denote the N data samples that have been collected from the real world, e.g., by asking people to write digits on a paper, capturing the image and processing it. The statistical mind believes there to be a data generating process, and consequently views these recorded images as outcomes of an underlying probability distribution. Hence, if we denote S as the bigger domain of all possible digits that can be written in reality, then $\mathcal{R} \sim S$. G is trained on $\mathcal{R}$ to approximate the underlying distribution by mapping noise z, sampled from a known latent space, into a generated image. During test time, sampling a new, unseen z and passing it through the trained G to create a new image can conceptually be understood as collecting a new, unseen handwritten digit by a new person from the real world; a digit that was not originally present in $\mathcal{R}$. In other words, the generated images, represented as $\mathcal{F}$, should ideally be able to function as if the original real data collection process has been repeated, i.e., $\mathcal{F} \sim S$.

3.1 Deep neural network classifiers for i.i.d detection

To study this, we need a metric $d(\mathcal{R}, \mathcal{F})$ that gives a clear signal whenever $\mathcal{R}, \mathcal{F} \overset{\text{iid}}{\sim} S$ regardless of the dataset type. FID, while generally helpful, is not the most appropriate metric in this case. To understand why, consider the dataset $\mathcal{R}$ being randomly shuffled and split in two halves $\mathcal{R}_1$ and $\mathcal{R}_2$. We know, by definition, that $\mathcal{R}_1, \mathcal{R}_2 \sim S$. If we set $\mathcal{R}$ as MNIST or FFHQ, FID($\mathcal{R}_1, \mathcal{R}_2$) will be different in magnitude. Hence there will be no way for us to confidently infer that $\mathcal{R}_1$ and $\mathcal{R}_2$ are actually iid samples in both cases. Hence, taking motivation from the line of work in image forensics community (Wang et al., 2020; Ojha et al., 2024), and especially from Lopez-Paz & Oquab (2017), we train various types of deep neural networks for binary classification to distinguish images from $\mathcal{R}$ and $\mathcal{F}$. If $\mathcal{R}, \mathcal{F} \sim S$, there should not be *any* signal available for a classifier to use to separate them,

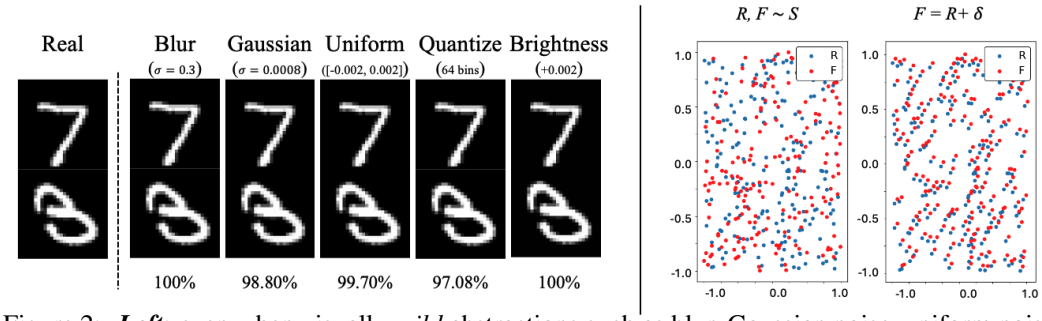


Figure 2: ***Left***: even when visually *mild* abstractions such as blur, Gaussian noise, uniform noise, quantization, or brightness shifts are applied, binary neural classifiers can still separate real and fake MNIST samples with high test accuracy. ***Right***: toy examples contrasting the desired i.i.d. case, $\mathcal{R}, \mathcal{F} \sim S$, with a non-i.i.d. shifted fake distribution, $\mathcal{F} = \mathcal{R} + \delta$, where the two sample sets are visually close but statistically structured differently.

at least on the test set. That is, the classifier's accuracy should only hover at $\sim 50\%$ throughout the training process. Specifically, we split $\mathcal{R}$ and $\mathcal{F}$ into their respective training and test splits and train a classifier, h, on $\mathcal{D}_{train} = \{\mathcal{R}_{train} \cup \mathcal{F}_{train}\}$ and test it on $\mathcal{D}_{test} = \{\mathcal{R}_{test} \cup \mathcal{F}_{test}\}$.

While $\mathcal{R}, \mathcal{F} \sim S$ necessitates $\sim 50\%$ test accuracy of h on $\mathcal{D}_{test}$, the inverse, i.e., chance performance on $\mathcal{D}_{test}$ does not necessarily imply $\mathcal{R}, \mathcal{F} \sim S$. There are two reasons for this. First, just because a single classifier, with its specific training configuration (e.g., network architecture, hyperparameters, weight initialization), cannot learn to separate $\mathcal{R}$ and $\mathcal{F}$ does not imply that a decision boundary does not exist. It is possible that a different h could have found it. We provide a practical workaround (though, not a theoretical one) by training a family of neural net classifiers $\mathcal{H}$ and observing if any of them can get $> 50\%$ test accuracy.

Since neural networks have generally been used as classifiers when there is a visible semantic difference in the two classes of images (e.g., ImageNet, CIFAR), there might be some questions whether they can also be used in cases when only very subtle differences exist between the two classes. To study this, we design an experiment where we take MNIST as $\mathcal{R}$ and create different corrupted versions, $\mathcal{R}'$s, of it by (i) very slightly blurring it ($\sigma = 0.3$), (ii) adding small Gaussian noise ($\sigma = 0.0008$), (iii) quantizing the images into 64 bins (original had 256), (iv) altering just a few pixels with small salt and pepper noise. The resulting images are shown in Fig. 2 left, where we see that there is barely any difference between real and corrupted versions of it. For each case, we train ResNet (He et al., 2016) on each $\mathcal{D}_{train} = \{\mathcal{R}_{train} \cup \mathcal{R}'_{train}\}$ and test on the corresponding $\mathcal{D}_{test}$. The results are shown at the bottom of Fig. 2, where see that the classifier consistently gets $> 90\%$ test accuracy. This stress test gives us some confidence that if there is a pattern to distinguish two classes of images, regardless of how subtle it is, at least some $h \in \mathcal{H}$ will be able to find it. More formally, the metric that we were looking for can be defined as following:

$$d_{\mathcal{H}}(\mathcal{R}, \mathcal{F}) = \max_{h \in \mathcal{H}} |\mathbb{E}_{\mathcal{R}}[h(X)] - \mathbb{E}_{\mathcal{F}}[h(X)]| \tag{1}$$

$d_{\mathcal{H}}(\mathcal{R}, \mathcal{F})$ will be zero if and only if none of the classifiers $h \in \mathcal{H}$ can learn to separate $\mathcal{R}$ and $\mathcal{F}$.

However, even if $\mathcal{H}$ could contain all possible classification functions, $d_{\mathcal{H}}(\mathcal{R}, \mathcal{F}) \sim 0$ still cannot guarantee $\mathcal{R}, \mathcal{F} \sim S$. Figure 2 (right) explains why. Case (a) depicts an when all the samples across $\mathcal{R}$ and $\mathcal{F}$ are i.i.d, whereas in case (b) $\mathcal{F} = \mathcal{R} + \delta$ where $\delta \sim 0$, i.e., not independent. In both cases, no function h can perform well on the respective $\mathcal{D}_{test}$. The question of quantitatively differentiating case (a) from (b) is an open one, and we leave it as such. Fortunately, as we will see in the results, we do not have to deal with this scenario for the most part. If the metric defined in Eq. 1 is positive, we can be sure that $\mathcal{R}$ and $\mathcal{F}$ are not iid samples from S.

3.2 Search for i.i.d through data abstractions

We know from prior work that generative models can introduce errors for various reasons (Zhang et al., 2019; Odena et al., 2016). A common example is checkerboard artifact due to the upsampling

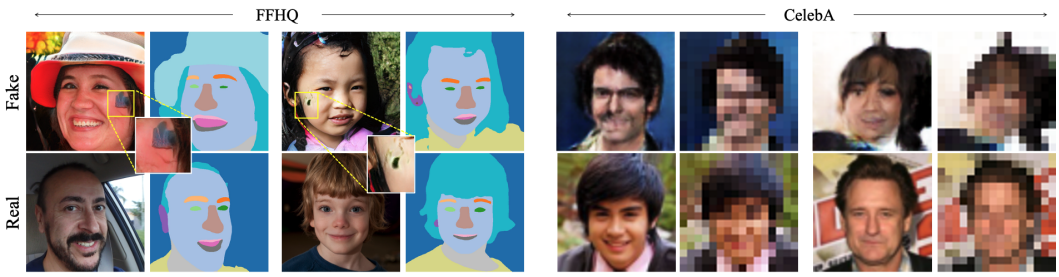


Figure 3: Motivation for image abstractions. Left: fake FFHQ images may contain localized artifacts, such as unnatural skin, that disappear after mapping the image to a higher-level semantic segmentation representation. Right: CelebA examples show a spatial abstraction, where downsampling removes fine image details while preserving coarse facial structure. We apply such abstractions to both real and fake images and test whether classifiers can still distinguish $f(\mathcal{R})$ from $f(\mathcal{F})$.

process. Beyond these, the images can often contain some semantic anomalies (e.g., weird lips, presence of six fingers) (Chai et al., 2020). But at the same time those images, notwithstanding the errors, appear good enough in other aspects, and also novel.

Could it be the case that $\mathcal{R}, \mathcal{F} \overset{\text{iid}}{\sim} S$ *if some small errors in* $\mathcal{F}$ *could be ignored?*

To study this, we propose to create abstracted versions of the original RGB real/fake images so that some information gets lost, with the hope of also washing away the errors by the generative model. Figure 3 explains the intuition. Taking the example of FFHQ, we notice that the original fake image has a weird artifact on the face, something that is not expected to be found in a real image. As an example abstraction, we extract segmentation masks of both real and fake images. In this space, that artifact is no more as that information is abstracted away. There are many other ways to do this as well. In the context of images, it is possible to preserve the resolution and apply operations such as blurring, adding noise, quantization, converting to edge/semantic maps etc. We can also reduce the dimensionality by operations such as downsizing, cropping. All of these provide avenues to hide the errors of G.

Formally, let f denote the function which transforms an image X into its abstracted version $f(X)$. After each such function application on both $\mathcal{R}$ and $\mathcal{F}$, we train a new collection of classifiers $\mathcal{H}$ from scratch. Extending the definition in Eq. 1, our generalizable metric becomes:

$$d_{\mathcal{H},f}(\mathcal{R}, \mathcal{F}) = \max_{h \in \mathcal{H}} |\mathbb{E}_{\mathcal{R}}[h(f(X))] - \mathbb{E}_{\mathcal{F}}[h(f(X))]| \tag{2}$$

In our experiments, we will search over the space of f to see if $d_{\mathcal{H},f}$ ever becomes ~ 0.

4 Experiments

We now explain the specific experiments we conduct to test if/when $\mathcal{R}, \mathcal{F} \overset{\text{iid}}{\sim} S$. In total, we consider four datasets of real images of increasing complexity & resolution; MNIST LeCun (1998) (28×28), CelebA Liu et al. (2015) ($64 \times 64 \times 3$), FFHQ Karras et al. (2019) (we consider $256 \times 256 \times 3$ version) and GenImage Zhu et al. (2023) ($256 \times 256 \times 3$). For each of these datasets, we gather corresponding fake images by either training appropriate generative models ourselves or by sampling from official pre-trained models.

4.1 MNIST

The real dataset, $\mathcal{R}$, is categorical in nature, having a total of 70k images distributed among the training ($\mathcal{R}_{train}$), validation and testing $\mathcal{R}_{test}$ splits of 50k, 10k and 10k images respectively. Each class ($0-9$) has the same number of images. We train ACGAN Odena et al. (2017) on $\mathcal{R}_{train}$, whose latent space consists of two inputs, a continuous noise vector $z \sim \mathcal{N}(0, 1)$ and a categorical code $c \sim \text{Cat}(K = 10, p = 1/10)$. We pass these inputs the generator to obtain a fake image $G(z, c)$.

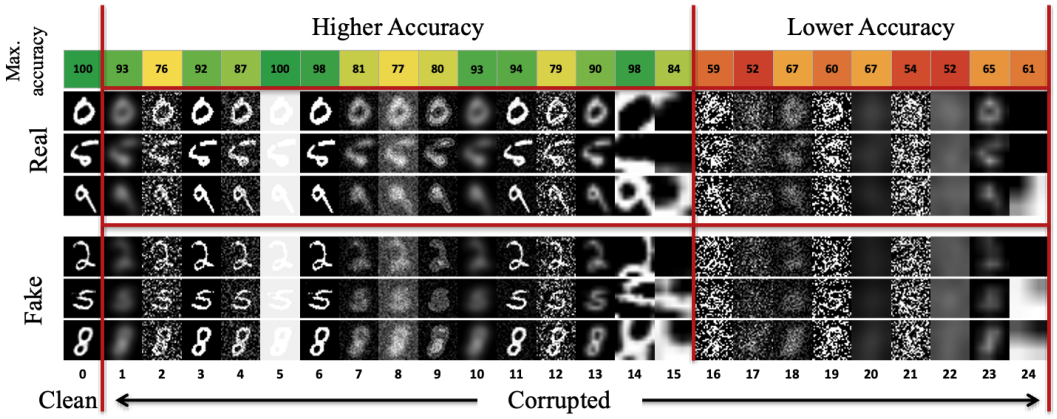


Figure 4: MNIST real-vs-fake detection under image abstractions. Columns show clean and corrupted versions of matched real and ACGAN-generated digits, while the heatmap reports the best test accuracy across DenseNet-Small, Mini-Xception, and ResNet-Small. Real and fake images remain separable through many abstractions and approach chance only when the digit content is heavily destroyed. Please see Table 6 for corruption configurations.

For each c (representing a digit category), we generate 5000 images by sampling that many z. This makes sure that $|\mathcal{R}_{train}| = |\mathcal{F}_{train}|$. Similarly, we generate $\mathcal{F}_{test}$ ensuring $|\mathcal{R}_{test}| = |\mathcal{F}_{test}|$. Our $\mathcal{H}$ consists of three models; ResNet, DesnseNet Huang et al. (2017) and mini-Xception Chollet (2016). We train all these classifiers h on $\mathcal{D}_{train} = \{\mathcal{R}_{train} \cup \mathcal{F}_{train}\}$ for 25 epochs, and test it on $\mathcal{D}_{test} = \{\mathcal{R}_{test} \cup \mathcal{F}_{test}\}$. This process is repeated separately for each abstraction level.

We consider the following abstraction operations f: (i) identity function (original real/fake images), (ii) blurring (with different σ), (iii) adding Gaussian/uniform/salt and pepper noise, (iv) quantization, (v) downsampling, (vi) cropping. In Fig. 4, we show many combinations of these, and the details of these configurations are present in Table 6. We also show the test accuracy across ResNet, DesnseNet and mini-Xception.

We begin the observation from left, i.e., original images, and see that all classifiers can easily separate them, which indicates the presence of at least some errors. This echoes the general finding of image forensics community. However, some errors seem to persist as we continue to apply different types and levels of abstraction. For example, even after blurring $\mathcal{R}$ and $\mathcal{F}$ with $\sigma = 5.0$, one of the $h \in \mathcal{H}$ is still able to separate them 80% of the time. Another example is downsizing the image to a 5×5 matrix, and a classifier can still identify $2/3^{rd}$ of the images. Note that all of these operations are abstracting away information in different ways. This means that the nature of error left in $\mathcal{F}$ (e.g., blurring vs downsizing), which gets picked up by h, is different.

On the right side of Fig. 4, we show those cases of f where all the classifiers perform only at around chance performance ($50 - 55\%$). The key thing to notice is the effect these f's have on the original image. The semantic content seems almost completely lost to the human eye. In other words, to have $\mathcal{F}$ become statistically indistinguishable from $\mathcal{R}$, one needs to, e.g., blur the image to such an extent, or make its resolution so small, or add so much noise, that the original digit in either the real or fake image is barely visible. We will discuss the implications of this later on in the paper.

4.2 FFHQ

Next, we experiment with the dataset whose release, along with the associated generative model, led to a genuine confusion among humans in being able to recognize what is real or fake. FFHQ comprises of 70k high quality images of human faces. We consider the 256×256 resolution version of the dataset and experiment with three different types of generative models trained on this dataset – StyleGAN2 Karras et al. (2020), VD VAE Child (2021) and latent diffusion models (LDM) Rombach et al. (2022a). From each of the generative model, we generate 70k fake images at the same resolution without any truncation, thus making $|\mathcal{R}| = |\mathcal{F}|$. This dataset is split into $80/20$ train/test split. We

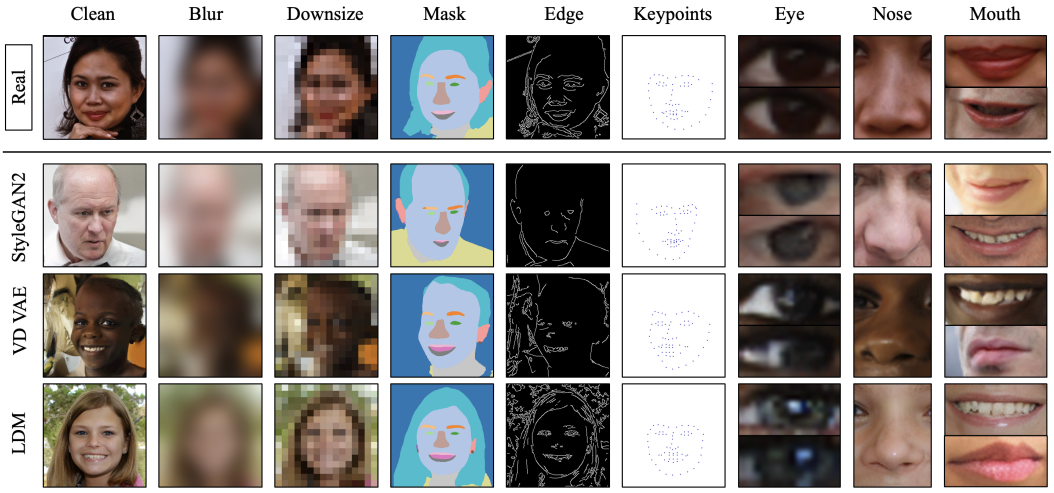


Figure 5: FFHQ real (top) and StyleGAN2, VD VAE, LDM-generated faces under various data abstractions. Even after abstracting away enough information from real and fake images, e.g., by converting them into semantic mask, we do *not* observe statistical indistinguishability, as a classifiers trained from scratch are able to obtain accuracies way above 50% in Table 1.

Model	Downsizing				Cropping				Blurring			
	64×64	16×16	4×4	1×1	64×64	16×16	4×4	1×1	σ=5	σ=10	σ=20	σ=40
StyleGAN2	99.8%	86.2%	58.4%	52.6%	100%	99.8%	96.3%	54.8%	98.0%	93.0%	74.0%	63.0%
VD VAE	98.9%	91.9%	69.3%	57.0%	100%	99.8%	85.9%	57.8%	95.8%	93.2%	83.0%	73.6%
LDM	99.5%	73.7%	58.1%	54.7%	100%	99.6%	82.5%	55.5%	93.0%	79.6%	64.8%	59.1%

Table 1: FFHQ real-vs-fake test accuracy of the best performing classifier under various data abstractions for StyleGAN2, VD VAE and LDM. Detection remains high for many spatially degraded inputs and approaches chance only under the strongest abstractions, such as 1×1 spatial collapse.

use ResNet, DenseNet and ViT as $\mathcal{H}$ and train them separately for (i) FFHQ vs StyleGAN2, (ii) FFHQ vs VD VAE, and (iii) FFHQ vs LDM.

We begin by considering three types of abstractions which can vary in strength; (i) blurring, (ii) downsizing and (iii) cropping. Fig. 5 (columns 1-3) shows the effect, and Table 1 shows the results. *To the human eye*, blurring ($\sigma = 20$) both real and fake images, or downsizing them to 16×16 resolution might appear to have removed all differences between the two. However, the classifier's performance remains way above $\sim 50\%$ for all the generative models. In other words, some unexpected signal still remains that the classifier uses to separate the two classes. In fact, to arrive at a state where the classifier cannot find much difference ($50 - 55\%$), we need to abstract to such an extent that the contents of the original image are virtually completely gone; e.g., downsizing/cropping to 1×1 region. Please see Fig. 11 in the appendix for a more detailed qualitative visualization of these abstraction processes.

	Edge	Mask	Keypoint	Eye	Nose	Mouth	L-eye	R-brow	R-ear
StyleGAN2	91.7%	76.1%	78.3%	99.8%	99.9%	99.9%	99.2%	98.9%	99.0%
VD VAE	95.0%	70.3%	76.5%	99.9%	99.9%	99.9%	99.9%	99.9%	99.9%
LDM	99.8%	67.8%	75.0%	99.9%	100.0%	100.0%	99.9%	99.9%	99.9%

Table 2: FFHQ real-vs-fake test accuracy of best performing classifier for StyleGAN2, VD VAE and LDM across various data abstractions (first 3 columns) and cropped face parts (last 6 columns).

Beyond these operations, we experiment with other abstractions which are semantically more meaningful given the high resolution nature of images. First, we use Canny edge detection to obtain the edge boundaries of real & fake images, which masks away, among other things, color information. Second, we convert the RGB image into its semantic mask using a pretrained segmentation model, something which conceptually is going to hide the high frequency noise, and potentially also hides

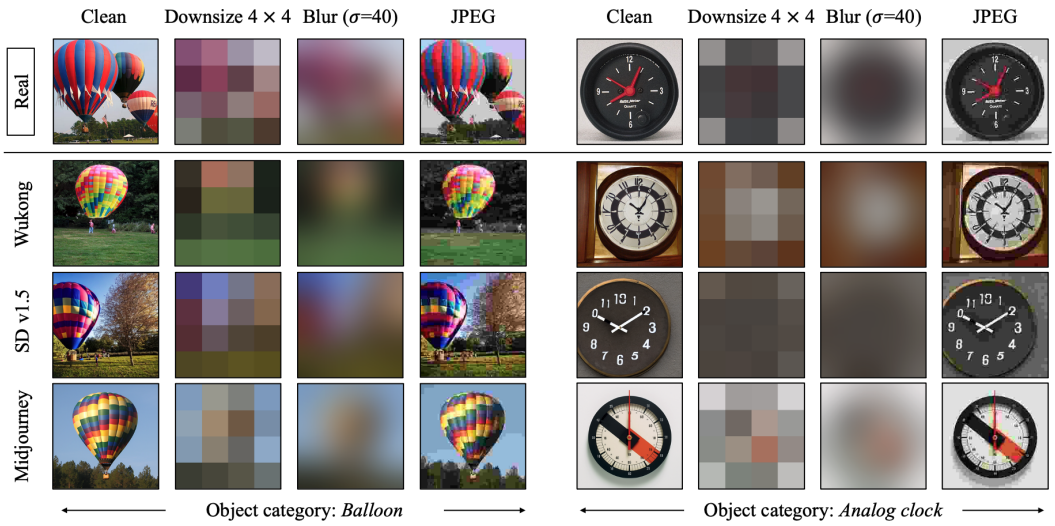


Figure 6: Real (top) and fake images (bottom three) from GenImage dataset under different spatial and blur abstractions. The original images are of 256×256 resolution. Even after abstracting the information content to a great extent by downsizing, blurring or compressing, the resulting real and fake images are still distinguishable by a classifier trained from scratch, as shown in Table 3.

Model	Downsizing			Cropping			Blurring			JPEG		
	16×16	4×4	1×1	16×16	4×4	1×1	$\sigma = 10$	$\sigma = 20$	$\sigma = 40$	50	25	5
Wukong	90.9%	81.1%	66.7%	97.5%	66.2%	61.9%	91.4%	87.2%	83.9%	96.6%	96.0%	92.6%
SD v1.4	91.1%	80.9%	68.6%	76.1%	61.5%	58.7%	91.1%	87.5%	82.4%	97.5%	94.8%	75.9%
SD v1.5	89.7%	80.2%	67.8%	75.6%	61.2%	58.4%	89.9%	86.0%	81.2%	96.9%	95.8%	91.1%
Midjourney	78.4%	69.1%	58.7%	93.2%	73.5%	57.4%	81.2%	74.7%	70.3%	92.7%	90.2%	84.3%

Table 3: GenImage real-vs-fake test accuracy under downsampling, cropping, blurring and JPEG abstractions. Detection remains well above chance for most abstractions and approaches chance only under extreme spatial collapse, such as 1×1 downsampling or cropping.

semantic anomalies as explained in Fig. 3. Next, we use a pretrained keypoint detector to extract 68 landmarks from a real/fake face image. This operation goes beyond the previous two in masking any explicit outline/edge in an image. The output from these operations (semantic mask, edge map, 68 landmark locations) gets fed into a classifier which is then trained to separate the real and fake class. Lastly, using the same semantic masks, we crop out certain regions of the face, e.g., its eyes, mouth, nose, ear, for both real and fake images. This way, we can use the classifiers in $\mathcal{H}$ to distinguish between real and fake eyes, or real and fake mouth etc. More details about the data format for semantic mask, edge map, landmark locations and details about the cropping mechanism is explained in the appendix Sec. B.3.

Figure 5 (columns 4-9) shows examples of the semantic and local-part abstractions, and Table 2 summarizes the classification results. In summary, none of these operations are enough to bring the test accuracy of all the classifiers to $\sim 50\%$. The implication is that the edges of fake images are different than real ones, fake eyes are different from real ones, the locations of various keypoints of fake images are different from real images etc. Similar to Table 1, this trend holds more or less similarly for all the three models. This principle of needing to abstract away almost the entire information of the original image to achieve statistical indistinguishability is observed also for another dataset of human faces – CelebA Liu et al. (2015), for which we present the results in appendix Sec. B.2.

4.3 GenImage

Finally, we experiment with more real world settings going beyond single object categories of handwritten digits or human faces. GenImage Zhu et al. (2023) was proposed for the task of fake image detection, and consists of real images from ImageNet Deng et al. (2009) and fake images

generated by popular text-to-image models. For our experiments, we consider images from four models; Wukong wuk (2022), Stable Diffusion (SD) v1.4 and v.15 Rombach et al. (2022b) and Midjourney Midjourney (2023). The real image dataset has $\sim 168,000$ images roughly evenly distributed across the 1000 object categories of ImageNet. Each of the fake image datasets also has the same number of images across the same categories. We use DenseNet as our classifier and train it separately for each generative model for binary classification, i.e., (i) ImageNet vs Wukong, (ii) ImageNet vs SD v1.4, (iii) ImageNet vs SD v1.5, and (iv) ImageNet vs Midjourney. The resolution of images is 256×256. For data abstraction, we borrow the operations described in Sec. 4.2 who strength can be varied – Downsizing, Cropping, Blurring. We add JPEG compression to this since it gets used in real world often. More details on GenImage dataset in appendix Sec. B.4.

Fig. 6 shows sample real and fake images from two object categories and the effect of different data abstraction operations (samples from more categories are shown in Fig. 12). The results of training the classifiers on these settings is shown in Table 3. Similar to the trend that we have seen so far, the test accuracy remains consistently high across many cases. The reason this should be surprising is because according to common understand, the errors or artifacts of the generative model manifest primarily in the high frequency components of a generated image Zhang et al. (2019); Wang et al. (2020). If so, an operation like downsizing to 4×4 or extremely high blurring ($\sigma = 40$) or very high JPEG compression (columns 2, 3, 4 respectively) should eliminate the artifact, which should result in statistical indistinguishability, i.e., test accuracy of $\sim 50\%$. However, our results suggest that each time the classifier can pick up some new signal to distinguish the two classes. Among the four models studied, Midjourney is the best (lower overall test accuracy). Collectively, these results across all of the datasets point to an understanding of fake images that we believe has been missing so far. We discuss this in the next section.

4.4 Analysis

We will focus on faces (FFHQ) since that has been the scope of much discussion in the community, but the general conclusions, we believe, apply to other domains such as MNIST and GenImage as well. What exactly is it that makes an image fake? Different works in the past have offered different perspectives. A line of work has attributed the fakeness to artifacts introduced due to the upsampling process. Within the context of images, the authors in Chai et al. (2020) show that *hair, weird background* of the face is the source for fakeness.

However, while these can certainly be legitimate sources of artifacts, our results seem to suggest that the nature of artifacts extends much beyond these isolated factors. The errors of G cannot be said to reside only in the high frequency checkerboard artifacts; because $\mathcal{R}$ and $\mathcal{F}$ differ in their semantic maps too. The errors also cannot reside only in the inability to model color details properly, because $\mathcal{R}$ and $\mathcal{F}$ differ in their (colorless) edge maps too. For faces, the errors can also not be contained just in the hair of the subject or the distorted background, because the eyes/mouth/nose of $\mathcal{R}$ and $\mathcal{F}$ differ too. In fact, the errors cannot just be some semantic anomaly in some part, because $\mathcal{R}$ and $\mathcal{F}$ differ even if we blur or downsize (e.g., 16×16) them exceedingly. The general conclusion from our experiments is that it is very difficult to find a *meaningful* aspect of the fake image within which it can be considered identical with real counterpart; i.e., $\mathcal{F}, \mathcal{R} \sim S$. The word 'meaningful' is important because them being statistically indistinguishable only when downsized to a single pixel, or after adding so much noise that the original signal seems lost to the human eye – is not very helpful.

5 Limitations

Our work primarily relies on classifier test to study whether perfection can be achieved by generative models. However, as explained in Fig. 2, while an accuracy of $\sim 50\%$ by multiple classifiers can hint at statistical indistinguishability, it still does not prove that $\mathcal{R}, \mathcal{F} \sim S$. Hence, right now we do not properly understand those cases where $\mathcal{R}$ and $\mathcal{F}$ do indeed become indistinguishable. Another limitation concerns the potential usage for our method to rank different generative models. For example, model A can be considered better than model B if A achieves statistical indistinguishability for a data abstraction sooner than B. However, computing such ranking will be much more computationally heavy than other available metrics like FID. Finally, our work has shown results on single-object datasets but not for cases where both real and fake images depict more complex scenes (e.g., traffic).

AI USE STATEMENT

In this work, we used generative AI tools for generating synthetic datasets, implementing methods, supporting qualitative and thematic data analysis. We have not used generative AI tools for designing or providing feedback on research methodology or experiments, interpreting results. Helping develop theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, proposing or refine hypotheses, assisting with translation, clean and reformat dataset are not applicable to this work. Additionally, we used generative AI tools for creating or editing software code, draft parts of the appendix section (the main paper has no AI usage). We have reviewed all AI-assisted work. The main parts of the code generated by AI has been verified by human authors for their accuracy. Some parts of the AI-generated code has been verified by multiple authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

We believe that our work puts forward a good news for the image forensics community because it calms the apparent threat by generative models. As we have explained throughout the paper, if it was true that generative models, apart from some small mistakes here and there, were getting close to perfection, then there might have been a genuine cause for worry because after those small errors get fixed, there will be nothing one could have done to verify whether some digital content is real or AI-generated. However, our work hints that these models are much further away from the natural data than what was previously imagined.

REPRODUCIBILITY STATEMENT

In this work, the learning algorithm are a collection of binary classifiers. The main ingredient is what the data is on which these classifiers are trained on. To that end, we have included the details of how the real and fake images are gathered for each datasets in the beginning of Sec. 4.1, 4.2 and 4.3, including their number, sampling process etc. The details of the data abstraction are given in Sec. B.1.4, B.2, B.3, B.4. The details about the classifier are given in Table 5, 7 and 9. We will open source the code upon publication.

REFERENCES

Wukong. https://xihe.mindspore.cn/modelzoo/wukong, 2022.

Ahmed Alaa, Boris Van Breugel, Evgeny S. Saveliev, and Mihaela van der Schaar. How faithful is your synthetic data? Sample-level metrics for evaluating and auditing generative models. In *Proceedings of the 39th International Conference on Machine Learning*, 2022.

Belhassen Bayar and Matthew C. Stamm. Constrained convolutional neural networks: A new approach towards general purpose image manipulation detection. *IEEE Transactions on Information Forensics and Security*, 13(11):2691–2706, 2018. doi: 10.1109/TIFS.2018.2825953.

Lucy Chai, David Bau, Ser-Nam Lim, and Phillip Isola. What makes fake images detectable? understanding properties that generalize. In *ECCV*, 2020.

Rewon Child. Very deep vaes generalize autoregressive models and can outperform them on images. In *ICLR*, 2021.

François Chollet. Xception: Deep learning with depthwise separable convolutions. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1800–1807, 2016. URL https://api.semanticscholar.org/CorpusID:2375110.

Davide Cozzolino, Justus Thies, Rossler Andreas, Riess Christian, Nießner Matthias, and Luisa Verdoliva. Forensictransfer: Weakly-supervised domain adaptation for forgery detection. In *arXiv*, 2019.

Davide Cozzolino, Giovanni Poggi, Riccardo Corvi, Matthias Nießner, and Luisa Verdoliva. Raising the bar of ai-generated image detection with clip, 2024. URL https://arxiv.org/abs/2312.00195.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

Alexander Diel, Tania Lalgi, Isabel Carolin Schröter, Karl F. MacDorman, Martin Teufel, and Alexander Bäuerle. Human performance in detecting deepfakes: A systematic review and meta-analysis of 56 papers. *Computers in Human Behavior Reports*, 2024.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, NIPS'14, pp. 2672–2680, Cambridge, MA, USA, 2014. MIT Press.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CVPR*, 2016.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, Gunter Klambauer, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NIPS*, 2017.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *CVPR*, 2020.

Terro Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *arXiv*, 2013.

Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. *CoRR*, abs/1904.06991, 2019.

Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

Ziqiang Li, Jiazhen Yan, Ziwen He, Kai Zeng, Weiwei Jiang, Lizhi Xiong, and Zhangjie Fu. Is artificial intelligence generated image detection a solved problem? In *NeurIPS*, 2025.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *ICCV*, 2015.

David Lopez-Paz and Maxime Oquab. Revisiting classifier two-sample tests. In *ICLR*, 2017.

Zeyu Lu, Di Huang, Lei Bai, Jingjing Qu, Chengyue Wu, Xihui Liu, and Wanli Ouyang. Seeing is not always believing: Benchmarking human and model perception of ai-generated images. In *NeurIPS*, 2023.

Long Ma Ma, Zhiyuan Yan, Jin Xu, Yize Chen, Qinglang Guo, Zhen Bi, Yong Liao, and Hui Lin. From specificity to generality: Revisiting generalizable artifacts in detecting face deepfakes. In *NeurIPS*, 2025.

Sara Mandelli, Diego Vila-Portela, David Vázquez-Padín, Paolo Bestagini, and Fernando Pérez-González. Beyond spectral peaks: Interpreting the cues behind synthetic image detection. In *ICASSP*, 2026.

Midjourney. Midjourney. https://www.midjourney.com/, 2023.

Augustus Odena, Vincent Dumoulin, and Chris Olah. Deconvolution and checkerboard artifacts. *Distill*, 2016. doi: 10.23915/distill.00003. URL http://distill.pub/2016/deconv-checkerboard.

Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. In *arxiv*, 2017.

Utkarsh Ojha, Yuheng Li, and Yong Jae Lee. Towards universal fake image detectors that generalize across generative models, 2024. URL https://arxiv.org/abs/2302.10174.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *ICLR*, 2016.

Anirudh Sundar Rajan, Utkarsh Ojha, Jedidiah Schloesser, and Yong Jae Lee. Aligned datasets improve detection of latent diffusion-generated images. In *ICLR*, 2025.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *ICML*, 2021.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. In *arXiv*, 2022.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Bjorn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022a.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022b.

Mehdi S. M. Sajjadi, Olivier Bachem, Mario Lučić, Olivier Bousquet, and Sylvain Gelly. Assessing generative models via precision and recall. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *NIPS*, 2016.

Ayush Sarkar, Hanlin Mai, Amitabh Mahapatra, Svetlana Lazebnik, D.A. Forsyth, and Anand Bhattad. Shadows don't lie and lines can't bend! generative models don't know projective geometry...for now. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 28140–28149, June 2024.

George Stein, Jesse Cresswell, Rasa Hosseinzadeh, Yi Sui, Brendan Ross, Valentin Villecroze, Zhaoyan Liu, Anthony L Caterini, Eric Taylor, and Gabriel Loaiza-Ganem. Exposing flaws of generative model evaluation metrics and their unfair treatment of diffusion models. In *NeurIPS*, 2023.

Sheng-Yu Wang, Oliver Wang, Richard Zhang, Andrew Owens, and Alexei A Efros. Cnn-generated images are surprisingly easy to spot...for now. In *CVPR*, 2020.

Zhendong Wang, Jianmin Bao, Wengang Zhou, Weilun Wang, Hezhen Hu, Hong Chen, and Houqiang Li. Dire for diffusion-generated image detection. In *2023*, 2023.

Zhiyuan Yan, Yong Zhang, Xinhang Yuan, Siwei Lyu, and Baoyuan Wu. Deepfakebench: A comprehensive benchmark of deepfake detection. In *NeurIPS*, 2023.

Andrii Yermakov, Jan Cech, Jiri Matas, and Mario Fritz. Deepfake detection that generalizes across benchmarks. In *WACV*, 2026.

Zu Zhang, Svebor Karaman, and Shih-Fu Chang. Detecting and simulating artifacts in gan fake images. In *WIFS*, 2019.

Mingjian Zhu, Hanting Chen, Qiangyu Yan, Xudong Huang, Guanyu Lin, Wei Li, Zhijun Tu, Hailin Hu, Jie Hu, and Yunhe Wang. Genimage: A million-scale benchmark for detecting ai-generated image, 2023.

APPENDIX

A COMPUTE RESOURCES

All experiments were run on a high-performance computing cluster using the SLURM workload manager. Training jobs were submitted primarily to the `Quick` GPU partition. Each training run used a single scheduled GPU, 4 CPU cores, and 16 GB of host RAM unless otherwise noted. The available GPU pool includes NVIDIA A100, A40, Titan RTX, Titan X, Titan V, and GTX 1080 Ti nodes; the exact GPU model was assigned by the cluster scheduler and was not pinned per experiment. All reported runs fit within the memory of a single GPU. Datasets, checkpoints, logs, and generated images were stored on the cluster data filesystem.

Resource	Description	Usage
Compute environment	Institutional HPC cluster managed with SLURM.	On-premise HPC
Accelerators	NVIDIA GPU nodes scheduled from the pool, including A100, A40, Titan RTX, Titan X, Titan V, and GTX 1080 Ti GPUs.	1 GPU per run
CPU and memory	Standard training allocation used by the reported experiments.	4 CPU cores, 16 GB RAM
Storage	Cluster data filesystem used for datasets, checkpoints, generated samples, logs, and analysis outputs.	personal env
Software environment	Conda environment with PyTorch and torchvision; jobs launched through SLURM scripts.	Python/PyTorch

Table 4: Compute environment used for the experiments.

Individual training runs were lightweight: MNIST classifier runs completed within the short `Quick` partition allocation using a single GPU, while the larger image-generation and face-dataset experiments were also run as independent single-GPU jobs. The total compute for the experiments reported in the paper is estimated to be on the order of tens of GPU-hours. The full research project consumed more compute than the final reported experiments because it included preliminary controls, architecture searches, corruption sweeps, failed or superseded runs, and pilot GAN experiments that were used to refine the final experimental design.

B TRAINING DETAILS

All experiments involving CelebA and FFHQ dataset use binary cross-entropy loss (BCELoss), an 80/20 train/test split, random seed 42, and a ReduceLROnPlateau learning rate scheduler (factor $= 0.5$, patience $= 5$). Labels are real $= 0$, generated $= 1$. All models output $P(\text{real}) \in [0, 1]$ via a sigmoid activation. Images are normalized to $[-1, 1]$ via $(x_{\text{uint8}}/255 - 0.5)/0.5$.

B.1 MNIST EXPERIMENTAL DETAILS

B.1.1 ACGAN TRAINING AND FAKE DATASET CONSTRUCTION

The MNIST generator is an auxiliary-classifier GAN (ACGAN) conditioned on the digit class. The generator input is a 100-dimensional latent vector $z \sim \mathcal{N}(0, I)$ and a categorical digit label $c \in \{0, \ldots, 9\}$. The class label is embedded into a 100-dimensional vector and multiplied element-wise with z before being passed through the generator.

The generator maps the conditioned latent code through a fully-connected layer followed by two upsampling convolutional blocks and a final `tanh` layer producing 28×28 grayscale images. The discriminator is a four-stride-2 convolutional stack with two heads: an adversarial real/fake head trained with binary cross-entropy and an auxiliary digit-class head trained with cross-entropy. Both networks are optimized with Adam (learning rate 2×10^{-4}, $\beta_1 = 0.5$, $\beta_2 = 0.999$), batch size 64, until the visual quality of generated images saturates (~ 200 epochs in practice).

For downstream real-vs-fake detection, fakes are generated class-conditionally. For each digit class d we sample independent latent vectors and generate enough fake images so that, for every split $s \in \{\text{train}, \text{eval}, \text{test}\}$,

$$|\mathcal{F}_{s,d}| = |\mathcal{R}_{s,d}|.$$

The resulting binary datasets are

$$\mathcal{D}_s = \{(x, 0) : x \in \mathcal{R}_s\} \cup \{(x, 1) : x \in \mathcal{F}_s\},$$

where label 0 denotes real and label 1 denotes fake. Both real and fake images are stored on disk as 8-bit PNGs and reloaded before training, so any sub-pixel float-precision differences from the generator are removed before either class enters the pipeline. Abstractions are applied *after* the split is constructed and identically to both classes (Sec. B.1.4).

B.1.2 Image-Space Classifiers

The main MNIST experiments use three binary classifiers: ResNet-Small, DenseNet-Small, and Mini-Xception. Each takes a single-channel image as input and outputs one sigmoid probability $h(x) \in [0, 1]$, interpreted as $P(x \text{ is fake})$. Models are trained from scratch separately for each abstraction level. Architectures are summarized in Table 5.

Model	Architecture	Parameters
ResNet-Small	3×3 convolution stem with 16 channels, followed by three residual stages: two 16-channel residual blocks, two 32-channel residual blocks with stride-2 downsampling in the first block, and two 64-channel residual blocks with stride-2 downsampling in the first block. Each residual block uses two 3×3 convolutions with batch normalization and ReLU, plus an identity or 1×1 projection shortcut. Final adaptive average pooling and a linear sigmoid head.	174,945
DenseNet-Small	DenseNet-BC style network with growth rate $k = 12$, initial 3×3 convolution to 24 channels, three dense blocks with four dense layers per block, and transition layers between blocks. Each dense layer uses BN–ReLU–1×1 convolution followed by BN–ReLU–3×3 convolution, then concatenates features with the input. Final batch normalization, adaptive average pooling, and a linear sigmoid head.	101,479
Mini-Xception	Compact Xception-style network. Entry block: $1 \to 8$ channels via a 3×3 convolution, batch normalization, and ReLU. Three residual depthwise-separable blocks with channel sizes 16, 32, and 64; the first two downsample by a factor of 2. Final adaptive average pooling and a linear sigmoid head.	13,089

Table 5: Binary classifier architectures used for the MNIST real-vs-fake abstraction experiments.

B.1.3 Classifier Optimization

All image-space classifiers are trained with binary cross-entropy:

$$\mathcal{L}(h) = -\frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{(x,y)\in\mathcal{D}_{\text{train}}} \big[y \log h(x) + (1 - y) \log(1 - h(x))\big].$$

We use Adam (learning rate 10^{-3}, default β), batch size 64, random seed 42, and train for 25 epochs. A step learning-rate scheduler is applied per epoch with multiplicative decay factor $\gamma = 0.7$, so the learning rate at epoch t is $\eta_t = 10^{-3} \cdot 0.7^{t-1}$ (this brings the LR to roughly 1.3×10^{-7} by the end of training, effectively freezing the network in the last few epochs). Inputs are normalized using the canonical MNIST mean and standard deviation, $(0.1307, 0.3081)$, after applying the abstraction operator. At test time, predictions are thresholded at 0.5, and we report the overall test accuracy as well as the per-class (real vs. fake) accuracy.

B.1.4 ABSTRACTION OPERATORS

Each abstraction operator f is applied to both real and fake images before classifier training and evaluation. Stochastic operators (Gaussian noise, uniform noise, salt-and-pepper) draw an independent noise sample per image using a deterministic per-split RNG (seeded by 42), so that every clean image has a single corrupted twin and the procedure is exactly reproducible.

PNG-roundtrip simulation. After applying any abstraction operator f, both the clean and corrupted images are quantized back to 8-bit precision before normalization:

$$\tilde{x} = \frac{\mathrm{round}(255 \cdot \mathrm{clip}(x, 0, 1))}{255}.$$

This step is applied identically to the real branch (which is already 8-bit on disk so the operation is a no-op there) and to the corrupted branch. Without it, sub-pixel float-precision differences introduced by f would survive into the network input even though they could never be observed in any actual saved PNG; we treat such artifacts as "ghost signals" that should not count as detectable.

The full preprocessing pipeline is therefore:

$$x \xrightarrow{f} x_f \xrightarrow{\text{8-bit round}} \tilde{x}_f \xrightarrow{\text{normalize}(\mu,\sigma)} \hat{x}_f,$$

where $(\mu, \sigma) = (0.1307,\ 0.3081)$ is the canonical MNIST normalization.

Operator definitions. The individual abstraction operators used in the experiments are defined below.

- **Identity:** $f(x) = x$.
- **Gaussian blur:** 2-D Gaussian convolution with bandwidth σ. The kernel size is $\max\big(3,\ (\lfloor 6\sigma + 0.5 \rfloor) \mid 1\big)$ to capture $\geq 99\%$ of the kernel mass and stay odd.
- **Gaussian noise:** $f(x) = \mathrm{clip}(x + \varepsilon, 0, 1)$ with $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ drawn independently per pixel.
- **Uniform noise:** $f(x) = \mathrm{clip}(x + \varepsilon, 0, 1)$ with $\varepsilon \sim \mathrm{Unif}(-a, a)$ drawn independently per pixel.
- **Classic Salt-and-pepper noise:** a fraction p of pixels (chosen by sampling $\lfloor 784\, p \rfloor$ pixel indices without replacement) is set to 0 or 1 with equal probability; remaining pixels are unchanged.
- **Soft salt-and-pepper noise:** a fraction p of pixels (chosen by sampling $\lfloor 784\, p \rfloor$ pixel indices without replacement) is perturbed by a fixed magnitude rather than forced to black or white. For each chosen pixel, we add either $+s$ or $-s$ with equal probability:

  $$f(x)_i = \begin{cases} \mathrm{clip}(x_i + s, 0, 1) & \text{w.p. } p/2, \\ \mathrm{clip}(x_i - s, 0, 1) & \text{w.p. } p/2, \\ x_i & \text{otherwise.} \end{cases}$$

  This is distinct from uniform noise, which adds a continuously-drawn perturbation to *every* pixel, and from classic salt-and-pepper, which forces selected pixels to extreme values regardless of the pixel's original content.
- **Quantization:** pixel values are reduced to L discrete levels, $f(x) = \mathrm{round}\big(x(L-1)\big)/(L-1)$.
- **Brightness shift:** $f(x) = \mathrm{clip}(x + b, 0, 1)$ for signed shift $b \in \mathbb{R}$.
- **Sharpening (unsharp masking):** $f(x) = \mathrm{clip}\big(x + \alpha\,(x - g_{\sigma=1}(x)),\ 0, 1\big)$, where $g_{\sigma=1}$ is a 2-D Gaussian blur with $\sigma = 1$ (yielding a 7×7 kernel via the same kernel-size rule above) and $\alpha \geq 0$ is the sharpening amount. $\alpha = 0$ is identity; $\alpha = 1$ is moderate; $\alpha \geq 2$ produces noticeable edge enhancement.
- **Downsampling:** the image is bilinearly resized from 28×28 to N×N and then bilinearly upsampled back to 28×28 — i.e., a low-pass filter that keeps only the information representable at resolution N.

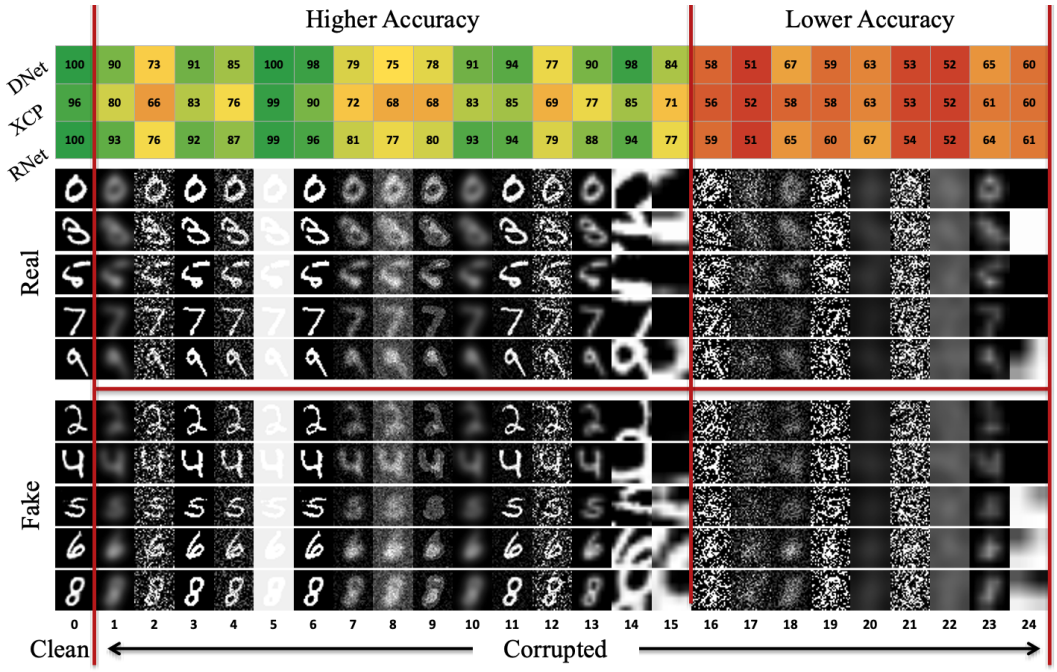


Figure 7: MNIST real-vs-fake detection under image abstractions. Columns show clean and corrupted versions of matched real and ACGAN-generated digits, while the heatmap reports test accuracy for DenseNet-Small (DNet), Mini-Xception (XCP), and ResNet-Small (RNet). Real and fake images remain separable through many abstractions and approach chance only when the digit content is heavily destroyed. Please see Table - 6 for corruption configurations.

- **Center crop:** the central $N \times N$ patch is retained, then bilinearly upsampled back to 28×28. This zooms into the central content rather than blacking out the periphery, so the classifier sees the same input shape with all available signal coming from the cropped region.
- **PCA:** each image is flattened to 784 dimensions, standardized using training-set statistics, projected onto the top k principal components, and classified directly in PCA space (Sec. B.1.6).

For mixed abstractions, operators are applied sequentially in the order written; for example, "blur 2.0 + quant 4 + gauss 0.1" denotes

$$f(x) = f_{\text{noise}}\big(f_{\text{quant}}\big(f_{\text{blur}}(x)\big)\big).$$

B.1.5 The 25 Abstraction Configurations

Table 6 lists the 25 configurations referenced by index in the corresponding figure. Configurations 0–15 appear in the "higher accuracy" (still detectable) panel; 16–24 appear in the "lower accuracy" (close to indistinguishable) panel.

B.1.6 PCA-Space Classifiers

For PCA experiments, real and fake images are flattened into 784-dimensional vectors. Standardization is fit only on the training split and then applied to the eval and test splits. PCA is also fit only on the standardized training data. We evaluate multiple reduced dimensions,

$$k \in \{2, 3, 4, 5, 6, 8, 10, 20, 50, 100, 200, 400, 600\}.$$

The PCA classifiers are trained with Adam (learning rate 10^{-3}), batch size 256, seed 42, and 25 epochs. The main PCA model is a one-dimensional residual MLP: a linear projection to 256 hidden units, three residual fully-connected blocks, and a final linear sigmoid head.

#	Operation
0	clean
1	blur σ=3.0
2	gauss σ=0.5
3	quant L=2
4	unif a=0.5
5	brightness b= + 0.95
6	sharpen α=5.0
7	blur σ=2.0 + gauss σ=0.1
8	bright b=0.3 + blur σ=2.0 + gauss σ=0.1
9	blur σ=2.0 + quant L=4 + gauss σ=0.1
10	sharp α=3 + blur σ=3
11	quant L=4 + gauss σ=0.1
12	quant L=2 + gauss σ=0.5
13	downsample N=10
14	crop N=10
15	crop N=5
16	unif a=1.0 + quant L=2
17	blur σ=5.0 + gauss σ=0.3
18	blur σ=3.0 + unif a=0.3
19	gauss σ=0.7 + quant L=2
20	blur σ=7.5
21	unif a=1.5
22	unif a=1.5 + blur σ=5.0
23	downsample N=5
24	crop N=2

Table 6: The 25 abstraction configurations used for the main MNIST experiment.

B.1.7 Controls and Implementation Notes

Real-vs-real negative control. We additionally train the same classifiers on a random $50{:}50$ split of $\mathcal{R}$ (i.e., real-vs-real, with no fake involved). Test accuracy stays at chance ($\approx 50\%$) across all three architectures and all abstraction operators, confirming that the real-vs-fake signal the detectors learn cannot be attributed to incidental split identity or class-imbalance shortcuts.

Digit-balanced fake splits. For real-vs-fake experiments, fake splits are digit-matched to the real splits: $|\mathcal{F}_{s,d}| = |\mathcal{R}_{s,d}|$ for every split s and digit class d. This prevents the classifier from exploiting any digit-frequency imbalance between real and fake.

Reporting. All reported test accuracies are computed on the held-out test split $\mathcal{R}_{\text{test}} \cup \mathcal{F}_{\text{test}}$, which is used neither for training nor for model selection. We report the final-epoch test accuracy after the 25-epoch schedule.

Reproducibility. All experiments use a fixed random seed of 42 for the abstraction RNG, model initialization, and the data shuffle. Models are trained on single NVIDIA GPUs of the institutional cluster; the full sweep of $\big|\mathcal{H}\big|\times$ (number of abstractions) jobs completes in under a few GPU-hours per architecture. All training, evaluation, and plotting code is released alongside the paper.

B.2 CelebA

Dataset. 202,599 real faces from CelebA paired with 202,599 DCGAN-generated faces (64×64 RGB). Training: 323,158 images; test: 81,040 images.

Ablations applied to both real and generated images:

- **Downsampling**: bilinear resize to $\{32, 16, 8, 4, 2, 1\}^2$ pixels.
- **Gaussian blur**: $\sigma \in \{6, 8, 10, 15, 20, 30\}$; image stays 64×64.
- **Center crop**: sizes $\{10, 4, 2, 1\}^2$ pixels extracted from the image center.

Training hyperparameters. 50 epochs, batch size 64. Baseline CNN uses Adadelta (lr $= 1.0$); all other architectures use Adam (lr $= 10^{-3}$). No weight decay.

Architectures (all adapted for 3-channel 64×64 RGB input):

Name	Optimizer	Params	Description
Baseline CNN	Adadelta 1.0	7,392,449	The model uses two convolutional layers, mapping 3 channels to 32 and then 32 to 64, followed by max pooling and dropout with probability 0.25. The flattened features pass through a 128-unit fully connected layer, dropout with probability 0.5, and a final one-unit output layer.
CNN+BatchNorm	Adam 10^{-3}	7,392,641	Baseline with BatchNorm after each conv; dropout removed.
AC-GAN Disc.	Adam 10^{-3}	99,937	4 stride-2 conv layers, LeakyReLU(0.2), BatchNorm, Dropout(0.25). Mirrors AC-GAN discriminator.
ResNet-Small	Adam 10^{-3}	175,233	3×3 stem; 3 residual blocks (16→32→64 ch, stride-2); AdaptiveAvgPool; FC(64→1).
Constrained CNN	Adam 10^{-3}	189,505	Zero-sum constrained first conv (5×5); 2 conv layers; AdaptiveAvgPool(4); FC(1024→128→1).
DenseNet-Small	Adam 10^{-3}	101,911	3 dense blocks, growth rate $k = 12$; AdaptiveAvgPool.
Mini-Xception	Adam 10^{-3}	13,233	Depthwise separable convolutions; AdaptiveAvgPool.
SRNet	Adam 10^{-3}	362,593	Steganalysis residual network; no early pooling to preserve high-frequency artifacts.

Table 7: Architectures used in the CelebA experiments.

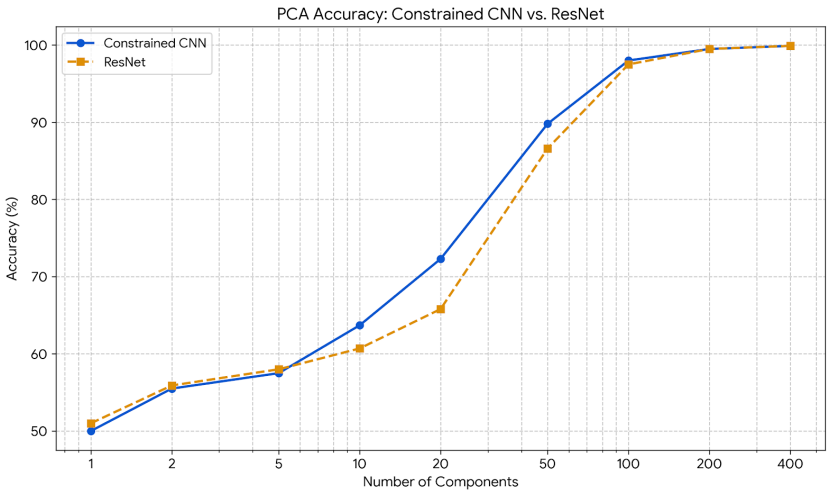


Figure 8: PCA results for CelebA/DcGAN configuration

We experiment with CelebA Liu et al. (2015), a slightly less complex version of FFHQ. Specifically, we use the cropped and aligned version which has $\sim$ 200k images at 64×64 resolution. This dataset, $\mathcal{R}$, is split between $\mathcal{R}_{train}$ and $\mathcal{R}_{test}$ having 160k and 40k images respectively. We train DcGAN Radford et al. (2016) on $\mathcal{R}_{train}$ for 10 epochs, whose latent input just consists of a single noise $z \sim \mathcal{N}(0, 1)$. Post training, we generate fake image datasets $\mathcal{F}_{train}$ and $\mathcal{F}_{test}$ having the same number of images as the real counterpart. In this case, out family of classifiers $\mathcal{H}$ includes ConstrainedCNN Bayar & Stamm (2018), DenseNet and SRNet. Similar to above, the classifiers are trained on $\mathcal{D}_{train}$ for 25 epochs and evaluated on $\mathcal{D}_{test}$. Furthermore, following the advice from Chai et al. (2020), we process the real and fake images to make sure that there are no differences due to formatting issues (e.g., one being saved in PNG and the other in JPEG). This way, the only differences that get discovered between $\mathcal{R}$ and $\mathcal{F}$ by $\mathcal{H}$ can safely be attributed to the errors of G.

We consider three types of abstractions: (i) downsizing, (ii) blurring, (iii) cropping. The results are shown in Fig. 9 and Table 8. . For each abstraction type, the strength of the operation (e.g., the amount of blur, crop) increases as we move from left to right. Generally speaking, the accuracy of the classifiers decreases along that trend. However, similar to MNIST, what is interesting is how often it stays above $> 50\%$. For example, even after downsampling $\mathcal{R}$ and $\mathcal{F}$ from 64×64 to 4×4, one of

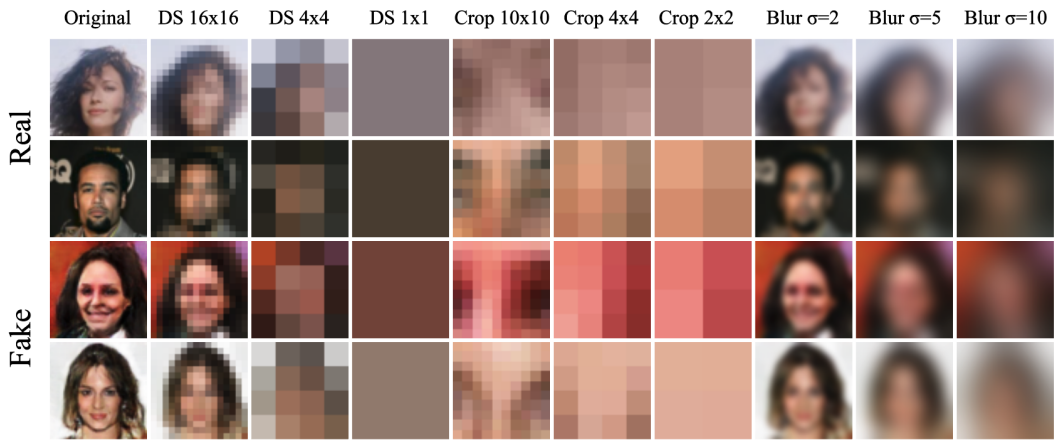


Figure 9: CelebA real and DCGAN-generated faces under spatial and blur abstractions.

Model	Downsizing				Cropping				Blurring			
	16×16	4×4	2×2	1×1	10×10	4×4	2×2	1×1	$\sigma = 6$	$\sigma = 10$	$\sigma = 20$	$\sigma = 30$
Constrained CNN	99.7%	90.2%	68.9%	53.9%	100.0%	99.9%	94.6%	57.1%	97.3%	91.7%	93.4%	96.2%
DenseNet	99.5%	89.0%	68.2%	54.0%	100.0%	100.0%	94.6%	57.1%	98.2%	91.0%	90.6%	91.2%
SRNet	99.8%	90.5%	68.4%	54.1%	100.0%	99.9%	94.5%	57.1%	96.4%	84.1%	83.6%	86.5%

Table 8: CelebA real-vs-fake test accuracy under downsampling, cropping, and blurring abstractions. Each entry reports binary classification accuracy on held-out real and DCGAN-generated test images. Detection remains well above chance for most abstractions and approaches chance only under extreme spatial collapse, such as 1×1 downsampling or cropping.

the h can still distinguish the two categories 90% of the time. A similarly surprising example is with blurring with $\sigma = 30$. That is so much blurring that it is hard to know if there was even a face in these images. So, it is safe to assume that all semantic anomalies, if they were present in any particular fake image, would have been removed. In spite of this, they are easily separable by all classifiers. We also show the results on PCA for two classifiers in Fig. 8. We observe that one can compress the data more than 200 times ($64 \times 64 \times 3 \rightarrow 50$) and still have a classifier separate $\sim 90\%$ of data points. The only cases where all the classifiers perform at around chance performance ($50 - 55\%$) are either if one downsizes the real/fake images to a single pixel or compresses the original signal more than 6000 times using PCA.

B.3 FFHQ

Dataset. 70,000 real FFHQ faces paired with 70,000 StyleGAN2/VD VAE/LDM-generated faces (256×256). Training: 112,000 images; test: 28,000 images. All models are trained for 50 epochs, batch size 64, weight decay 10^{-4}. CNN classifiers use Adam (lr $= 10^{-3}$); ViT uses Adam (lr $= 10^{-4}$).

Two families of inputs are evaluated. The first operates on **derived representations** rather than raw pixels:

- **Canny edge maps**: grayscale binary edge images computed with `cv2.Canny()`.
- **Face segmentation maps**: per-pixel class labels (0–18) produced by a pretrained BiSeNet (ResNet-18 backbone) fine-tuned on CelebAMask-HQ. Raw integer label maps are used as classifier input.

The second operates directly on **raw RGB pixels**, with the following ablations:

- **Center crop**: sizes $\{64, 16, 4, 1\}^2$ pixels.
- **Downsampling**: bilinear resize to $\{64, 16, 4, 1\}^2$ pixels.
- **Gaussian blur**: $\sigma \in \{5, 10, 20, 40\}$; image stays 256×256.

The same three architectures are used across both input families. For derived representations the input is 1-channel grayscale; for raw RGB the first convolution is adapted to accept 3 channels.

Name	Optimizer	Params (1ch)	Params (3ch)	Description
ResNet	Adam 10^{-3}	174,385	174,817	Stem Conv+BN+ReLU; 3 residual stride-2 layers (16→32→64 ch); AdaptiveAvgPool; FC(64→1).
DenseNet	Adam 10^{-3}	133,481	134,057	Stem Conv(stride=2)+MaxPool; 3 DenseBlocks (growth rate 16, 4 layers each); AdaptiveAvgPool; FC→Sigmoid.
ViT	Adam 10^{-4}	12,874,241	12,875,265	patch_size=16, dim=512, depth=6, heads=8, mlp_dim=1024, dropout=0.1.

Table 9: Architectures used in the FFHQ experiments. Parameter counts are given for 1-channel (derived representations) and 3-channel (raw RGB) inputs.

Per-face-part dataset construction. In addition to the global classifiers above, we train one binary classifier per face part on tight crops localized to that region. Crops are produced from the same 70k real FFHQ + 70k StyleGAN2 images in two passes: (i) a one-time bounding-box localization pass, and (ii) a fixed-size cropping pass driven by per-part templates derived from the localization statistics.

Bounding-box localization. We start from CelebAMask-HQ-style segmentation maps (one class label per pixel, 19 classes covering eyes, brows, nose, mouth, ears, hair, etc.), produced by the same BiSeNet parser used above for the segmentation-map experiments. For each of 14 target parts (eye, nose, mouth, mouth_inner, left/right_eye, brows, left/right_brow, left/right_ear, eye_g (glasses), neck, hair) we map the part to the relevant class IDs (e.g. *eye* = brows ∪ eyes, *nose* = class 10, *left_eye* = class 4, *eye_g* = class 6) and extract the pixel mask of those classes. A tight axis-aligned bounding box is then computed per image per part with two robustness adjustments:

- **Percentile-trimmed extents.** Instead of the full min/max of the mask, we use the 1st and 99th percentile on each axis (i.e. keep the central 98% of pixels). This suppresses parser hallucinations that occasionally place a few stray pixels far from the true region.
- **Largest connected component.** For laterally paired parts (*left/right_eye*, *left/right_brow*, *left/right_ear*) we additionally restrict the mask to the largest 4-connected component before computing the box, which discards scattered mis-classified pixels on the wrong side of the face.

A part is marked *excluded* when glasses (class 6) overlap the eyes (so eye crops would contain frames rather than skin), and *invalid* when fewer than 20 pixels of the target class are present. The resulting boxes are written to JSON per part, alongside a status flag, so downstream stages can read only the `ok` boxes without recomputing segmentation.

Per-part fixed crop templates. Per-image bounding boxes vary in size and aspect ratio, so we collapse them to one fixed crop size per part. For each part we collect the widths and heights of all `ok` boxes across both classes (real and fake) and pick the 98th percentile of each, rounded up to the nearest multiple of 8 (template $W_p \times H_p$). The choice of P98 is motivated empirically: the width distributions exhibit a knee around the 98th percentile, after which boxes grow primarily because of parser noise or extreme poses rather than because the part is genuinely larger. Rounding to a multiple of 8 makes the crop dimensions friendly for both CNN strides and ViT patch grids without introducing artificial padding pixels.

Cropping pass. Given the per-part template (W_p, H_p) and the per-image bounding box (x_0, y_0, x_1, y_1), we compute the bounding-box center (c_x, c_y) and carve out a window of size $W_p \times H_p$ centered on (c_x, c_y). If the window would fall outside the image we shift it inward (clamp), which preserves the template size at the cost of a small location offset for boundary cases (typical for ear crops near the image edge). No resampling, padding, or interpolation is applied: all crops are exact pixel sub-windows of the source image, so the real-vs-fake comparison cannot be biased by an asymmetric resampling pipeline. Crops are written as PNG with no metadata, again matching real and fake byte-for-byte except in pixel content.

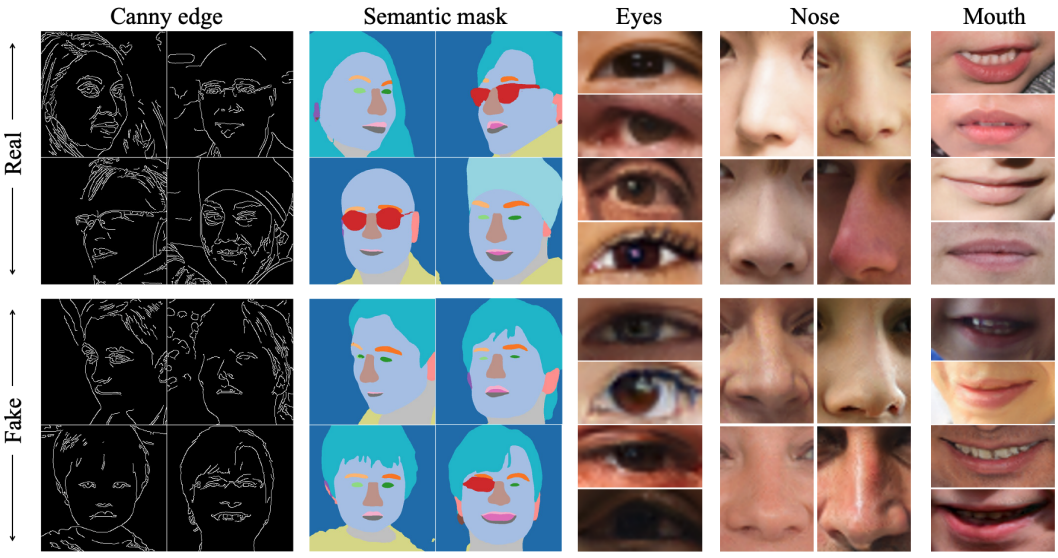


Figure 10: FFHQ real and StyleGAN2-generated faces under semantic and local-part abstractions.

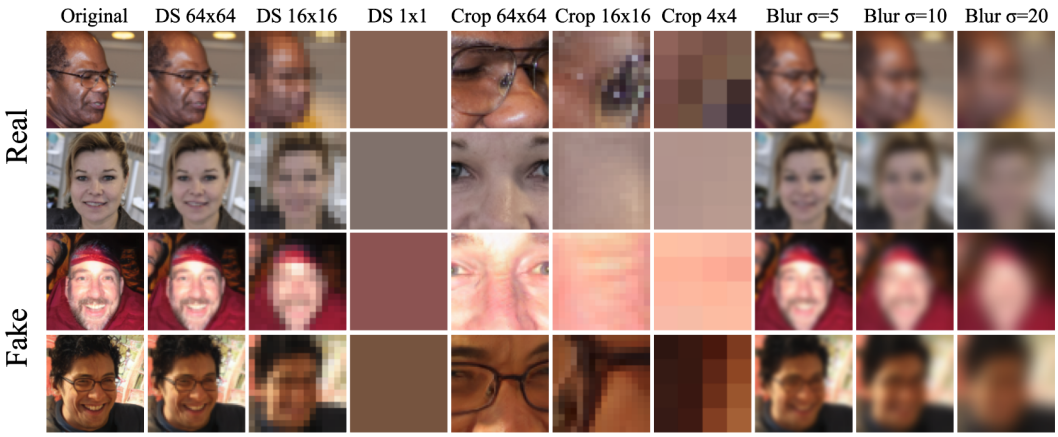


Figure 11: FFHQ real and StyleGAN2-generated faces under spatial and blur abstractions.

Sanity checks. Before training we verify that (a) for every part the cardinality of real and fake `ok` crops differs by less than 2%, (b) the file-size, image-resolution, and per-channel mean/std distributions of real and fake crops are statistically indistinguishable (logistic-regression baseline accuracy $\leq 55\%$ on each, all near chance), and (c) the train/val/test split (70/15/15, deterministic md5 hash of `image_id`) places every `image_id` in exactly one split, so a given source identity never appears across splits even when the same identity is cropped for multiple parts. After per-split class balancing (subsample the larger class to match the smaller, deterministic seed) the real and fake counts are exactly equal in every split for every part.

Per-part classifiers. The same three architectures from above (ResNet, DenseNet, ViT) are trained once per (part, architecture) pair on the 3-channel RGB crop. CNN architectures keep the 10^{-3} Adam recipe used for the global FFHQ classifiers; ViT switches to a tuned recipe for these small per-part crops, in which patch size is reduced from 16 to 4 (yielding $\sim 16\times$ as many tokens given the smaller spatial extent), AdamW with weight decay 0.05 replaces Adam, the learning rate is set to 5×10^{-4}, and a 5-epoch linear warm-up is followed by a cosine schedule. This change is necessary because the default Adam 10^{-3} + StepLR recipe leaves ViT at 63–72% test accuracy on these crops; the smaller-patch / AdamW recipe closes the gap to within 0.1–0.2% of the CNN baselines.

Model	Downsizing				Cropping				Blurring			
	64×64	16×16	4×4	1×1	64×64	16×16	4×4	1×1	σ=5	σ=10	σ=20	σ=40
ResNet	99.1%	67.2%	54.5%	52.6%	100.0%	99.7%	81.7%	54.7%	96.5%	76.9%	57.0%	55.2%
DenseNet	99.5%	77.6%	55.5%	52.5%	100.0%	99.8%	96.3%	54.8%	96.7%	82.3%	66.8%	59.6%
ViT	99.8%	86.2%	58.4%	51.3%	100.0%	99.8%	95.1%	50.5%	98.0%	93.0%	74.0%	63.0%

Table 10: FFHQ real-vs-fake test accuracy under downsampling, cropping, and blurring abstractions. Each entry reports binary classification accuracy on held-out real and StyleGAN2-generated test images. Detection remains high for many spatially degraded inputs and approaches chance only under the strongest abstractions, such as 1×1 spatial collapse.

	Edge	Mask	Eye	Nose	Mouth	L-eye	R-brow	R-ear
ResNet	82.4%	71.1%	99.69%	99.28%	99.85%	98.97%	98.59%	74.97%
DenseNet	85.9%	76.1%	99.84%	99.90%	99.71%	99.24%	97.19%	86.19%
ViT	91.7%	70.9%	99.83%	99.78%	99.77%	98.34%	98.88%	98.95%

Table 11: Real-vs-fake test accuracy on FFHQ vs. StyleGAN2 across derived representations (Edge, Mask) and six cropped face parts.

B.4 GENIMAGE

Dataset. We use the ImageNet-based GenImage benchmark Zhu et al. (2023), which pairs real ImageNet photographs (the `nature` split) with images produced by several text-to-image generators prompted with the 1,000 ImageNet class names ($\sim$162 generated images per class per generator). We evaluate four generators: Midjourney, Wukong, Stable Diffusion v1.4, and Stable Diffusion v1.5. For every generator we use the same pool of real images: all 167,701 `nature` images (161,701 train + 6,000 val) across all 1,000 classes.

Format and resolution matching. GenImage's real and generated images differ in native resolution, aspect ratio, and file format. The generated images are square (1024×1024 for Midjourney, 512×512 for Wukong and Stable Diffusion) and saved as PNG. The real ImageNet photographs are JPEGs with arbitrary aspect ratios (median $\sim 500\times375$). To prevent the classifier from exploiting these formatting differences Chai et al. (2020), all images are brought to a common representation before training. Resizing the real images straight to a square distorts their aspect ratio. We checked this by comparing grids of randomly sampled real and generated images from the same ImageNet class, and the distortion was visible. We therefore center-crop each real image to a square of side $\min(h, w)$, which preserves its aspect ratio, and then resize it to 256×256. Each generated image, which is already square, is resized directly to 256×256. Both resizes use Lanczos resampling, and every image in both classes is stored as a lossless 8-bit PNG. This single set of processed real images is shared across all four generators.

Splits. GenImage's official train and val folders are merged. Each generator provides more generated images than there are real images, so the larger class is subsampled to the size of the smaller one. The balanced pool is then split at random into 80% train and 20% test (seed 42). For Midjourney this yields 268,322 training and 67,080 test images, split equally between real and fake.

Ablations applied to both real and generated images:

- **Downsampling**: bilinear resize to $\{64, 16, 4, 1\}^2$ pixels, then bilinear upsampling back to 256×256.
- **Gaussian blur**: $\sigma \in \{5, 10, 20, 40\}$ at 256×256.
- **Center crop**: sizes $\{64, 16, 4, 1\}^2$ pixels taken from the image center, then bilinearly upsampled back to 256×256.
- **JPEG compression**: re-encoding at quality $q \in \{75, 50, 25, 5\}$ and decoding again before the image is passed to the network.

Every ablation is applied on the fly and identically to both classes, and the output always has the same $256\times256\times3$ input shape. A separate network is trained from scratch for each (generator, ablation, strength) combination.



Figure 12: Real (top) and fake images (bottom three) from GenImage dataset under different spatial and blur abstractions.

Training hyperparameters. We use the 3-channel DenseNet from the FFHQ experiments (Table 9; 134,057 parameters). It is trained with binary cross-entropy, Adam (lr $= 10^{-3}$, weight decay 10^{-4}), and a ReduceLROnPlateau scheduler (factor 0.5, patience 5) for 50 epochs with batch size 64. Inputs are normalized to $[-1, 1]$. Because the full data pool ($\sim$36 GB of PNGs) exceeds what the cluster's network file system can serve efficiently as individual files, the images are stored in tar shards and loaded into memory once per job. This changes only the I/O and has no effect on the data.

Real-vs-real control. For each generator's setup we also train the same DenseNet on a random 50/50 split of the real images, with no generated images involved. This checks whether the preprocessing pipeline itself gives the classifier an incidental cue.